Bottom-up Reuse Guidelines: Object-Oriented Programming

Bottom up Reuse Guidelines: Object-Oriented Programming

by Ryan Gerard (Innovim / NASA GSFC)

Based on the presentation, "Object Oriented Programming and its relevance to Software Reuse", by Ryan Gerard to the NASA Earth Science Data Systems Software Reuse Working Group, monthly telecon, December 20, 2006.

What are Objects?

- Software objects are conceptually similar to real-world objects: they too consist of a state and possible behaviors (methods).
- An object stores its state in fields (variables in some programming languages) and exposes its behavior through methods (functions in some programming languages).
- Methods operate on an object's internal state and serve as the primary mechanism for object-to-object communication.

Properties of Object Oriented Programming

- Modularity: The source code for an object can be written and maintained independently of the source code for other objects.
- Information-hiding: By interacting only with an object's methods, the details of its internal implementation remain hidden from the outside world.
- **Software Reuse**: If an object already exists (perhaps written by another software developer), you can use that object in your program. This allows specialists to implement/test/debug complex, task-specific objects, which you can then trust to run in your own code.
- Pluggability and debugging ease: "If a bolt breaks, you replace it, not the entire machine."

Inheritance is another property that can be applied to object-oriented programming. Objects can be derived from existing objects given that they have *some amount* in common with each other. With respect to reuse, complex objects can be implemented by inheriting from simpler objects (these most likely are already available).

Type Polymorphism

- **polymorphism** means allowing a single definition to be used with different types of data (specifically, different classes of objects)
- The ability of objects belonging to different types to respond to method calls of methods of the same name, each one according to an appropriate type-specific behavior.
- In practical terms, polymorphism means that if class B inherits from class A, it doesnt have to inherit everything about class A; it can do some of the things that class A does differently.

Relevance of Polymorphism to Reuse

- Polymorphism allows client programs to be written based only on the abstract interfaces of the objects which will be manipulated.
- Gives the programmer a means to "reuse" what is needed from a class and override what needs to be different.
- It is possible to reuse the same client program, while only tailoring the object.

References

- http://java.sun.com/docs/books/tutorial/java/concepts/
- http://en.wikipedia.org/wiki/Polymorphism_in_object-oriented_programming
- http://en.wikipedia.org/wiki/Polymorphism_%28computer_science%29